Unidad Aritmética y Lógica.

La unidad aritmética y lógica (ALU – Arithmetic Logic Unit) es la encargada de realizar las operaciones aritméticas (suma y resta) y lógicas básicas (AND, OR, XOR, NAND, NOR y XNOR). Esta unidad es la que realiza prácticamente el procesamiento en un microprocesador. Para realizar el diseño de la ALU partiremos del análisis de un circuito sumador y restador completo de un bit, para después generalizarlo para "n" bits. Dos esquemas son analizados para este circuito:

- Sumador y restador con acarreo en cascada.
- Sumador y restador con acarreo anticipado.

Análisis de un circuito sumador y restador con acarreo en cascada.

Se parte del análisis clásico, aquí en primera instancia, se genera la tabla de verdad para un sumador completo de 1 bit. Esta tabla contiene las entradas del bit A_i a sumar con el bit B_i y el acarreo de entrada C_i. Como salidas se tiene el resultado de la suma S_i y el siguiente acarreo C_{i+1} generado a ser tomado en cuenta por el bit más significativo siguiente o MSB (Most Significant Bit), como se muestra en la tabla 1.

Donde: $i = 0,1,\ldots,n-1$

n=número_de_bits_del_circuito_sumador

A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabla 1 Sumador completo de 1 bit.

Para obtener la ecuación de salida de S_i se utiliza una compuerta que de forma natural suma bits. Se trata de la compuerta lógica *OR* exclusiva (*XOR*), cuya tabla de verdad se muestra en la tabla 2.

A	*B*	*XOR*
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 2 Tabla de verdad de la compuerta XOR

Autor: Victor Hugo García Ortega

La ecuación para la salida *XOR* se muestra a continuación:

$$XOR = A\overline{B} + \overline{A}B = A \oplus B \qquad (1)$$

Por lo tanto, la ecuación para S_i queda así:

$$S_i = A_i \oplus B_i \oplus C_i \qquad (2)$$

Para obtener la ecuación de salida simplificada de C_{i+1} es posible manejar diferentes esquemas, el más común es usar un mapa de Karnaugh como se muestra en la figura 1.

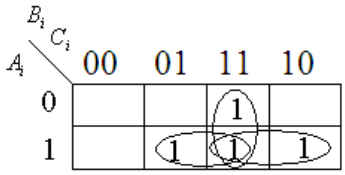

Figura 1 Mapa de Karnaugh para obtener la ecuación del acarreo.

Aplicando el procedimiento de simplificación se tiene la ecuación 3.

$$C_{i+1} = A_i C_i + B_i C_i + A_i B_i \qquad (3)$$

Empleando las ecuaciones 2 y 3 es posible generar un sumador de *n* bits conectando los acarreos en cascada. Aunque este esquema no es el mejor, es el más simple e intuitivo. En la figura 2 se muestra un ejemplo de conexión para un sumador de 4 bits usando los acarreos en cascada.

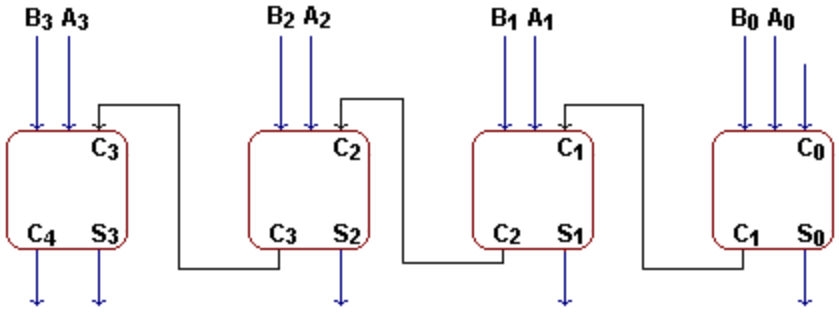

Figura 2 Circuito Sumador de 4 bits

En la figura 3 se muestra la operación de suma para 4 bits.

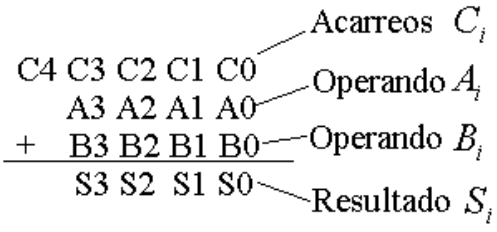

Figura 3 Operación de suma con 4 bits

Autor: Victor Hugo García Ortega

En la operación de suma de la figura 3 el acarreo inicial C_0 debe ser cero. Usando las ecuaciones 2 y 3 podemos escribir las ecuaciones para cada uno de los bits del circuito sumador de la figura 2.

$$S_0 = A_0 \oplus B_0 \oplus C_0 \quad (4)$$
$$C_1 = A_0 C_0 + B_0 C_0 + A_0 B_0 \quad (5)$$
$$S_1 = A_1 \oplus B_1 \oplus C_1 \quad (6)$$
$$C_2 = A_1 C_1 + B_1 C_1 + A_1 B_1 \quad (7)$$
$$S_2 = A_2 \oplus B_2 \oplus C_2 \quad (8)$$
$$C_3 = A_2 C_2 + B_2 C_2 + A_2 B_2 \quad (9)$$
$$S_3 = A_3 \oplus B_3 \oplus C_3 \quad (10)$$
$$C_4 = A_3 C_3 + B_3 C_3 + A_3 B_3 \quad (11)$$

El circuito de la figura 2 sólo puede realizar sumas, de números de 4 bits. Como se desea realizar un circuito sumador y restador, entonces es necesario analizar una operación de resta. Para ello se propone realizar la resta a través de la suma de la siguiente forma:

$$r = A - B = A + (-B) \quad (12)$$

La ecuación 12 nos dice que la resta es una suma donde necesitamos colocar a B en complemento a dos para representarlo en forma negativa.

Como ejemplo, vamos a suponer que $A = 5$ y $B = 3$, por lo que la resta estaría dada por:

$$r = A - B = A + (-B) = 5 + (-3) = 2 \quad (13)$$

De esta manera, se utiliza un circuito sumador **para realizar una sustracción** y los operandos son los que incorporan el bit de signo. Si la operación de sustracción se realiza en binario empleando un sumador de 4 bits, es necesario poseer el equivalente binario de los números 5 y -3. El equivalente binario del número 5 es 0101. Para representar el binario de -3 es necesario utilizar la representación en complemento a dos. Esta se obtiene colocando el número binario normal, después obtenemos el complemento a uno, que es simplemente negar los bits del número, y posteriormente sumar 1 al resultado del complemento a uno. Esto lo podemos observar en la figura 4.

0011 — número 3 en binario
1100 — complemento a uno del número 3

$C_4\ C_3\ C_2\ C_1\ C_0$
00000 — acarreos
1100
+0001 sumamos 0001 con el acarreo $C_0 = 0$ para obtener complemento a dos
1101 Este número representa al -3

La operación anterior es equivalente a:

$C_4\ C_3\ C_2\ C_1\ C_0$
00001 — acarreos
1100
+0000 sumamos 0000 con el acarreo $C_0 = 1$ para obtener complemento a dos
1101 Este número representa al -3

Figura 4 Complemento a dos del número 3

La operación de la ecuación 13 queda como se muestra en la figura 5.

$C_4\ C_3\ C_2\ C_1\ C_0$
11010 ⟵ Acarreos
5 0101 ⟵ número 5 en binario
+(-3) +1101 ⟵ número -3 en complemento a dos
2 0010 ⟵ resultado

La operación anterior es equivalente a:

$C_4\ C_3\ C_2\ C_1\ C_0$
11011 ⟵ Acarreos
5 0101 ⟵ número 5 en binario
+(-3) +1100 ⟵ número -3 en complemento a uno
2 0010 ⟵ resultado

Figura 5 Operación de resta

Como podemos observar en la figura 5 la operación de resta es en realidad una suma con dos modificaciones al operando B_i o sustraendo. Estas modificaciones son la negación o complemento a uno y posteriormente la suma de uno al complemento a uno. Este uno que sumamos al complemento a uno, lo podemos colocar en el acarreo inicial C_0 . Además de las modificaciones al sustraendo, debemos definir una señal de control que permita seleccionar entre las dos operaciones (suma o resta) a realizar con los datos. Esta señal la llamaremos *Binvert* y su función se muestra en la tabla 3.

Binvert	Operación
0	Suma
1	Resta

Tabla 3 Señal de selección de operación

Con esta señal de control ya podemos realizar las modificaciones al circuito sumador de la figura 2 para convertirlo en un circuito sumador y restador. La primer modificación al operando B_i o sustraendo es obtener su complemento a uno. Esto lo obtenemos con un circuito inversor, pero este circuito solo debe negar a B_i cuando se seleccione la operación de resta, en el caso de que se quiera realizar la operación de suma, **el operando no debe ser negado**. Esta selección se obtiene con un circuito multiplexor de dos entradas, una salida y un selector, donde la entrada 0 es B_i sin negar y la entrada 1 es $\overline{B_i}$. El selector es la señal de control $Binvert$. Estas modificaciones se muestran en la figura 6.

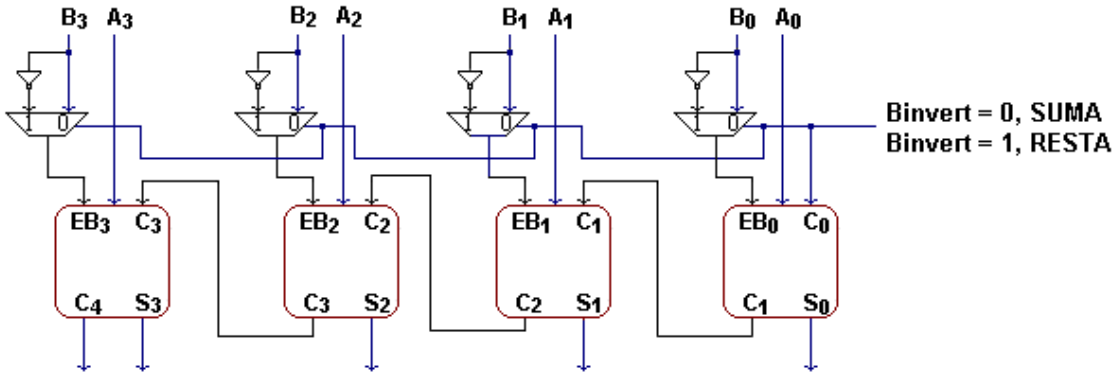

Figura 6 Circuito sumador y restador

Analicemos la figura 6. Si queremos realizar una operación de suma colocamos en la señal $Binvert$ un cero con esto seleccionamos la entrada 0 del multiplexor permitiendo a B_i pasar a la entrada EB_i del circuito sumador. Al mismo tiempo el acarreo inicial C_0 es colocado en cero obteniendo una operación de suma. Por el contrario, si queremos realizar una operación de resta colocamos en la señal $Binvert$ un uno, con esto seleccionamos la entrada 1 del multiplexor permitiendo a $\overline{B_i}$ (complemento a uno) pasar a la entrada EB_i del circuito sumador. Al mismo tiempo el acarreo inicial C_0 es iniciado con uno, obteniendo la segunda modificación del operando B_i y de esta forma el complemento a dos y la operación de resta.

Finalmente analicemos el circuito multiplexor de la figura 7, que es el circuito usado en la figura 6. Este es un multiplexor de dos entradas una salida y un selector.

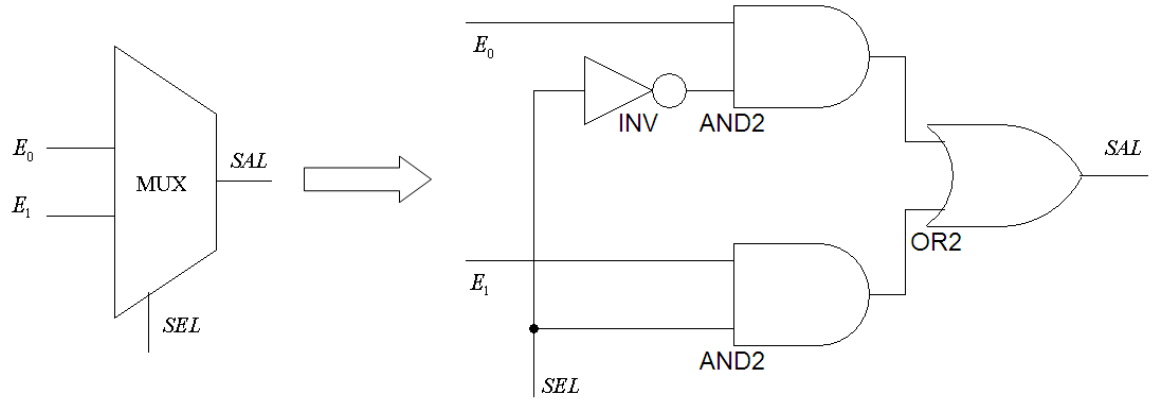

Figura 7 Multiplexor de dos entradas, una salida y un selector

La ecuación de salida del multiplexor mostrado en la figura 7 es la siguiente:

$$SAL = E_0 \overline{SEL} + E_1 SEL \qquad (14)$$

En el caso del circuito sumador y restador mostrado en la figura 6, la entrada E_0 es la entrada B_i, la entrada E_1 es la entrada $\overline{B_i}$, el selector es la señal $Binvert$ y la salida del multiplexor es la señal EB_i. Con estas equivalencias la ecuación 14 se puede reescribir como se muestra a continuación:

$$EB_i = B_i \overline{Binvert} + \overline{B_i} Binvert \qquad (15)$$

Al observar la ecuación 15 nos damos cuenta de que se trata de una operación XOR entre la señal B_i y $Binvert$ de acuerdo a la ecuación 1. Es decir, la ecuación 15 la podemos reescribir como:

$$EB_i = B_i \oplus Binvert \qquad (16)$$

Con esto nos damos cuenta que el multiplexor que estamos utilizando es en realidad una operación XOR. Por lo tanto, podemos sustituir el multiplexor mostrado en la figura 6, tal como se muestra en la figura 8.

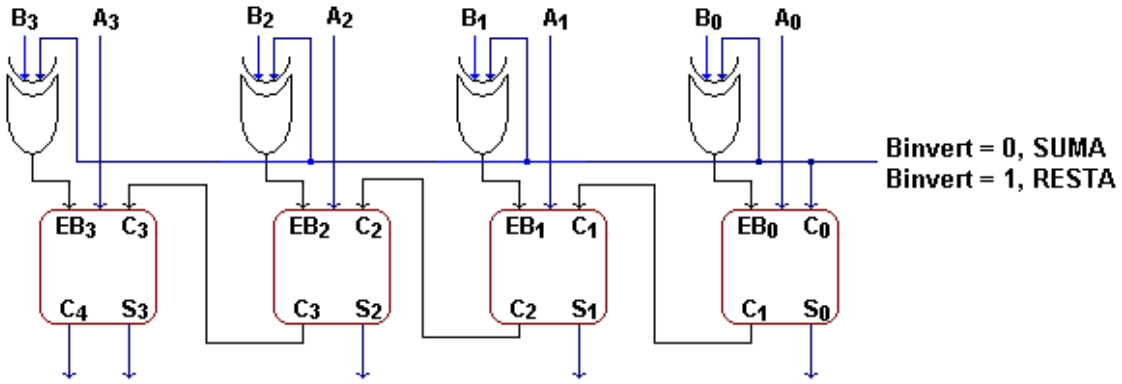

Figura 8 Circuito sumador y restador

Autor: Victor Hugo García Ortega

Finalmente las ecuaciones para el circuito sumador y restador son una modificación de las ecuaciones 2 y 3, donde se agrega la parte de la compuerta XOR. Esto nos da como resultado las ecuaciones 17 y 18.

$$S_i = A_i \oplus EB_i \oplus C_i \quad (17)$$

$$C_{i+1} = A_i C_i + EB_i C_i + A_i EB_i \quad (18)$$

Las ecuaciones 16, 17 y 18 son las ecuaciones a programar en el dispositivo lógico programable usando un lenguaje de descripción de hardware como VHDL, el ambiente de desarrollo a usar para este programa es ISE WEB PACK de la compañía Xilinx. De la figura 8 podemos observar que las señales $A_0,\ldots,A_3$, $B_0,\ldots,B_3$, C_0 y $Binvert$, son señales de entrada. Las señales $S_0,\ldots,S_3$ y C_4 son señales de salida. El tener la señal C_0 independiente de $Binvert$ nos permite poder conectar varios componentes en cascada para formar un circuito sumador y restador de mayor cantidad de bits.

Analicemos el desempeño del circuito de la figura 8. Una vez que se tiene el valor en la señal EB_i se necesitan de 2 retardos de propagación para obtener el valor del acarreo C_1 . Posteriormente el valor del acarreo C_2 se obtiene después de 4 retardos de propagación (2 de C_1 + 2 de C_2), el valor del acarreo C_3 se obtiene después de 6 retardos de propagación (2 de C_1 + 2 de C_2 + 2 de C_3) y el valor del acarreo C_4 se obtiene después de 8 retardos de propagación (2 de C_1 + 2 de C_2 + 2 de C_3 + 2 de C_4). En general para un circuito sumador y restador de n bits el valor del acarreo C_n se obtiene después de $2n$ retardos de propagación. Si estamos hablando que la ALU es el núcleo de procesamiento del procesador, este retardo afecta significativamente el desempeño de la ALU y por lo tanto del procesador. **Este esquema se conoce como circuito sumador y restador con acarreo en cascada**.

Análisis de un circuito sumador y restador con acarreo anticipado.

Este circuito elimina el retardo de $2n$ en el bit de acarreo C_n del circuito sumador y restador con acarreo en cascada. Para entender cómo se elimina dicho retardo analizaremos la ecuación de salida para C_{i+1} generada del mapa de Karnaugh mostrado en la figura 1, pero ahora agrupando como se muestra en la figura 9.

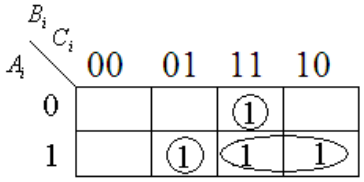

Figura 9 Mapa de Karnaugh para obtener la ecuación del acarreo.

Aplicando el procedimiento de simplificación se tiene la ecuación 19.

$$\begin{aligned} C_{i+1} &= A_i B_i + A_i \overline{B_i} C_i + \overline{A_i} B_i C_i \\ C_{i+1} &= A_i B_i + C_i (A_i \overline{B_i} + \overline{A_i} B_i) \\ C_{i+1} &= A_i B_i + C_i (A_i \oplus B_i) \end{aligned} \qquad (19)$$

Para analizar la ecuación anterior definiremos las ecuaciones mostradas en 20.

$$\begin{aligned} P_i &= A_i \oplus B_i \\ G_i &= A_i B_i \end{aligned} \qquad (20)$$

Con las ecuaciones anteriores la ecuación 19 se reescribe como se muestra en la ecuación 21.

$$C_{i+1} = G_i + C_i P_i \qquad (21)$$

Si se desarrolla la ecuación 21 para un sumador de 4 bits tenemos los acarreos de la ecuación 22.

$$\begin{aligned} C_0 &= acarreo_{inicial} \\ C_1 &= G_0 + C_0 P_0 \\ C_2 &= G_1 + C_1 P_1 = G_1 + (G_0 + C_0 P_0) P_1 \\ C_2 &= G_1 + G_0 P_1 + C_0 P_0 P_1 \\ C_3 &= G_2 + C_2 P_2 = G_2 + (G_1 + G_0 P_1 + C_0 P_0 P_1) P_2 \\ C_3 &= G_2 + G_1 P_2 + G_0 P_1 P_2 + C_0 P_0 P_1 P_2 \\ C_4 &= G_3 + C_3 P_3 = G_3 + (G_2 + G_1 P_2 + G_0 P_1 P_2 + C_0 P_0 P_1 P_2) P_3 \\ C_4 &= G_3 + G_2 P_3 + G_1 P_2 P_3 + G_0 P_1 P_2 P_3 + C_0 P_0 P_1 P_2 P_3 \end{aligned} \qquad (22)$$

De los acarreos de la ecuación 22 se puede observar lo siguiente:

- Al realizar la sustitución en cada ecuación C_{i+1} por el acarreo C_i, todas las ecuaciones resultantes dependen del acarreo inicial C_0.
- Se tiene un retardo de propagación constante de $2td$ para todos los acarreos.
- Se puede obtener una ecuación que genere las ecuaciones para cada bit de acarreo C_{i+1}. Para un sumador de n bits, donde $i=0,1,2,\ldots,n-1$ se obtiene la ecuación 23.

$$C_{i+1}=G_i+\sum_{j=0}^{i-1} G_j \prod_{k=j+1}^{i} P_k + C_0 \prod_{l=0}^{i} P_l \qquad (23)$$

Por lo tanto las ecuaciones completas para el sumador y restador con acarreo anticipado son:

$$EB_i = B_i \oplus Binvert$$
$$P_i = A_i \oplus EB_i$$
$$G_i = A_i EB_i$$
$$S_i = A_i \oplus \mathrm{EB}_i \oplus \mathrm{C}_i = P_i \oplus \mathrm{C}_i$$
$$C_{i+1}=G_i+\sum_{j=0}^{i-1} G_j \prod_{k=j+1}^{i} P_k + C_0 \prod_{l=0}^{i} P_l \qquad (24)$$

Banderas de estado

De cada operación se obtienen los valores de las banderas de acarreo (C), negativo (N), cero (Z) y desbordamiento (OV). Estas banderas son conocidas como **banderas de estado** puesto que proporcionan el estado de la ALU después de cada operación aritmética o lógica. Estas banderas son almacenadas en el registro de banderas o registro de estado para poder ser usadas con las instrucciones de comparación y brincos condicionales.

Análisis de la bandera de overflow.

A	B	C	D	SIN SIGNO	CON SIGNO
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	-8
1	0	0	1	9	-7
1	0	1	0	10	-6
1	0	1	1	11	-5
1	1	0	0	12	-4
1	1	0	1	13	-3
1	1	1	0	14	-2
1	1	1	1	15	-1

Tabla 4: Números con signo y sin signo de 4 bits.

El análisis de la bandera lo vamos a realizar analizando varios ejemplos de sumas con datos binarios de 4 bits con signo.

Sumas sin overflow	Sumas con overflow
0 0 0 0 0 C_i + 0 0 1 1 A = 3 0 1 0 0 B = 4 0 1 1 1 S=7	0 1 0 0 0 C_i + 0 1 0 0 A = 4 0 1 0 1 B = 5 1 0 0 1 S=-7
1 1 1 1 0 C_i + 1 1 0 1 A = -3 1 0 1 1 B = -5 1 0 0 0 S=-8	1 0 0 0 0 C_i + 1 1 0 0 A = -4 1 0 1 1 B = -5 0 1 1 1 S=7

Autor: Victor Hugo García Ortega

En las sumas sin overflow los últimos dos acarreos son iguales “00” ó “11”, en cambio, en las sumas con overflow los últimos dos acarreos son diferentes “01” ó “10”. Para poder detectar el overflow se usa una compuerta XOR cuyas entradas son los dos últimos acarreos.

Para el caso de una ALU de “n” bits, cada una de las banderas de la ALU tiene las siguientes funciones:

Bandera Z (zero). Esta bandera se pone en 1 cuando todos los bits del resultado de la ALU son cero, en caso contrario tiene 0. Esto se logra con una compuerta NOR que tiene como entradas todos los bits de resultado.

Bandera C (carry). Esta bandera muestra el valor que tiene el último acarreo de la ALU, es decir, el acarreo C_n es la bandera C.

Bandera N (negative). Esta bandera muestra el valor que tiene el bit más significativo (bit de signo) del bus del resultado, es decir, el bit n-1 de resultado es la bandera N. Cuando N tiene 1 significa que el resultado es negativo, de lo contrario es positivo.

Bandera OV (overflow). Esta bandera es usada para operaciones aritméticas con signo. Cuando OV = 1 marca un desbordamiento en el resultado, lo que significa que el resultado no se puede representar con los bits que tenemos para el resultado. Esto se logra con una compuerta XOR entre los dos acarreos más significativos, es decir, las entradas de la compuerta XOR es C_n y C_{n-1} .

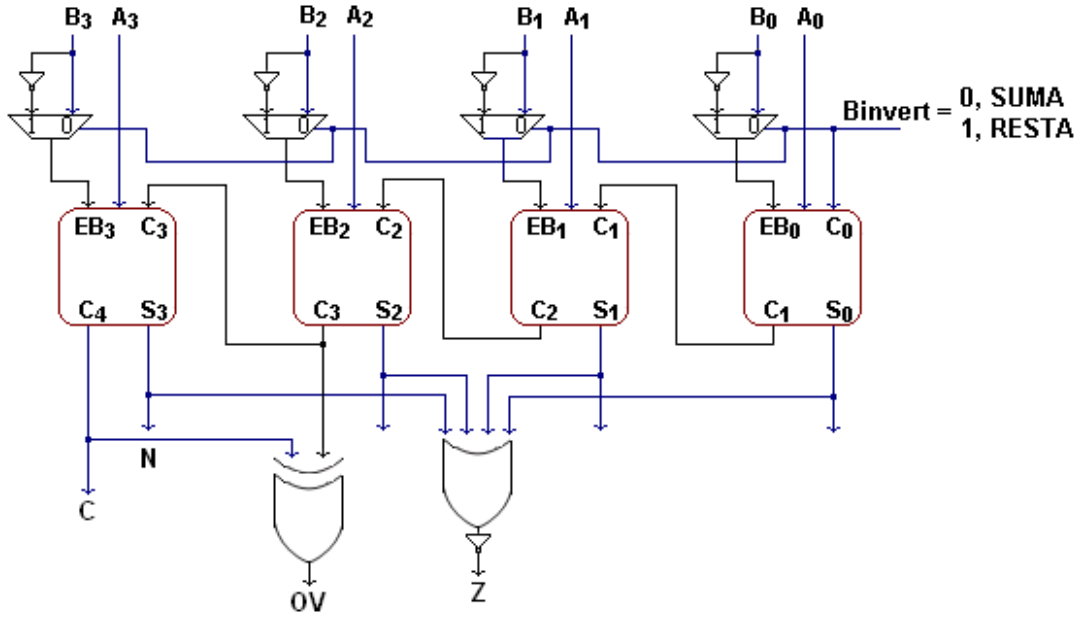

Figura 10: Circuito sumador y restador con banderas de estado.